Lecture 3 - Servlet Lifecycle & form submission

HttpServlet

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of HttpServlet must override at least one method, usually one of these:

1. doGet, if the servlet supports HTTP GET requests
2. doPost, for HTTP POST requests
3. doPut, for HTTP PUT requests
4. doDelete, for HTTP DELETE requests
5. init and destroy, to manage resources that are held for the life of the servlet
6. getServletInfo, which the servlet uses to provide information about itself

About deploying applications

The webapps directory is used to hosts the compiled application, tomcat then scans the webapps directory initially to cache them to the work directory where the actual application is deployed and run. Thus, when you recompile the application, tomcat may not load the latest changes – we need to restart the server. However, a new application can be cached, deployed and run automatically, since this new application wasn’t cached previously in work. This behavior occurs because of the *autoDeploy* parameter in server.xml.

An alternative to restarting the server is to create .war - archive file. To create this file use below command,

jar cf file\_name.jar . – this would archive all .class files as .jar file

where cf – create file

We use this utility to archive the JAVA EE file structure into a .war file, then set the unpackWARS in server.xml to true, to run the application as .war file. Tomcat automatically deploys the changed .war file without restarting the server. Similarly, if we delete the .war file the cached application would cease to run, as tomcat deletes the cached application as well.

This also solves the issues of deploying an application with multiple class by archiving them.

Lifecycle of a servlet:

1. Creates servlet object (constructor called by container – viz tomcat)
2. Calls the init() method (creates the process)
3. Servlet container creates request & response objects
4. Servlet container routes both objects to service() method (to determine the type of request)
5. Calls doGet() method for the received GET request
6. Calls destroy() method to delete the request & response objects

Global & Service variables

Information to be displayed or accessible publicly should be defined as a global variable. These are usually set in the init() call and is associated to service. User specific information must be defined as a service variable. These are usually set in the service() call and is associate to the thread.

Form submission

Assume the below form,

<FORM METHOD=’get’ ACTION=’servlet-url’?

User Name: <INPUT TYPE=’text’ NAME=’user’/><br/>

Password: <INPUT TYPE=’password’ NAME=’pswd’/><br/>

</FORM>

Using the form variables in the service() methods,

doGet(request,response){

String username = request.getParameter(“user”);

}